数理解析研究所講究録
第427巻 1981年 175-189

Subset Graphs and Their Applications to Databases

Yahiko Kambayashi
Department of Information Science
Kyoto University
Kyoto Japan

1. Introduction

A graph is a powerful tool to represent and manipulate binary relations. There are, however, many relations which cannot be represented by binary relations, and thus hypergraphs are used. When we want to solve database problems, hypergraphs seem to be not powerful enough. If we can distinguish several characteristics of relations, further applications of graph formulation will be possible. By this motivation, in this paper a new representation method of relations by subset graphs will be introduced together with their applications to database problems.

In subset graphs, each vertex corresponds to a subset of a given set S. Several kinds of edges (directed or undirected) are defined to represent different characteristics of relations. In general there will be $2^{|S|}-1$ vertices and thus it will be very difficult to manipulate the graph. Practically in many cases, however, existence of edges can be assumed to be not independent, and in such cases the reduction of computation is possible.

In Section 2, a subset graph is defined. Each vertex of the

graph corresponds to a set of events, and four kinds of edges, which show characteristics of relations, are used. These edges are, however, shown to be special cases of one kind of edges. Several basic properties are also shown. There is a strong similarity between properties of subset graphs and dependency theory in the relational databases[1]-[5][8][9].

Basic computation methods for subset graphs are discussed in Section 3. There exists a Boolean expression to solve problems on subset graphs. Some subclasses of general subset graphs are known to be manipulated efficiently.

In Section 4, as an example of labeled subset graphs, a new authorization mechanism for database systems is introduced.

In Section 5, other applications of subset graphs to databases are summarized.

2. Subset Graphs

A subset graph is defined as $G(S,V,E)$, where $S=\{A_1,A_2,\ldots,A_n\}$ is a set of events and each vertex in E corresponds to X_i which is a non-empty subset of S. There are the following four types of edges.

(1) $X_i \longrightarrow X_j$: If X_i occurs then X_j occurs.

(2) $X_i \rightarrow X_{j1}, \ldots, X_{jp}$: If X_j occurs then at least one of X_{jk} $(k=1,\ldots,p)$ occurs. It can be represent as $X_i \rightarrow X_{j1} | X_{j2} | \cdots | X_{jp}$.

(3) X_i —— X_j : At least one of X_i and X_j occurs.

(4) X_i --- X_j : At least one of X_i and X_j does not occur.

Here, 'X_i occurs' means that 'every event in X_i occurs.'

Some edges are not independent and thus we only need to write an essential part of the whole subset graph.

There are the following rules from the above definition.

P1: If X_i occurs then any subset X_j of X_i occurs.

For $X_i \supseteq X_j$, $X_i \rightarrow X_j$.

P2: If X_i does not occur then any superset X_j of X_i does not occur.

P3: The transitive rule is satisfied by edges of types (1) and (2).

If $X_i \rightarrow X_{j1} | X_{j2} | \cdots | X_{jp}$ $X_{j1} \rightarrow X_{k1} | \cdots | X_{kq}$,

then $X_i \rightarrow X_{k1} | \cdots | X_{kq} | X_{j2} | \cdots | X_{jp}$.

P4: The augmentation rule is satisfied by edges of types (1) and (2).

If $X_i \rightarrow X_{j1} | X_{j2} | \cdots | X_{jp}$ and $Y_i \supseteq Y_{jk} (k=1,\ldots,p)$,

then $X_i \cup Y_i \rightarrow X_{j1} \cup Y_{j1} | X_{j2} \cup Y_{j2} | \cdots | X_{jp} \cup Y_{jp}$.

P5: The decomposition rule is satisfied by edges of type (1).

If $X_i \rightarrow X_j$,

then for any $A \in X_j$, $X_i \rightarrow A$.

P6: The subset rule is satisfied by edges of types (1) and (2).

If $X_i \rightarrow X_{j1} | X_{j2} | \cdots | X_{jp}$ and $X_{jk} \supseteq Y_{jk} (k=1,\ldots,p)$,

then $X_i \rightarrow Y_{j1} | Y_{j2} | \cdots | Y_{jp}$.

P7: The subset rule is satisfied by edges of type (3).

If X_i —— X_j and $X_i \supseteq Y_i$, $X_j \supseteq Y_j$,

then X_i —— Y_j.

P8: The superset rule is satisfied by edges of type (4).

If X_i --- X_j and $X_i \subseteq Y_i$, $X_j \subseteq Y_j$,

then Y_i---Y_j.

P9: The following property is satisfied by edges of type (4).

If X_i---X_j

then for any Y_i and Y_j such that $X_i \cup X_j = Y_i \cup Y_j$

Y_i---Y_j.

Notice that edges of type (1) are similar to the functional dependencies in the relational database model and that edges of type (2) are similar to the multivalued dependencies[1]-[5][8][9]. In the database theory, althogh functional dependencies are special cases of multivalued dependencies, the former cannot be expressed precisely as a special form of the latter. In our model any edge of type (1) can be expressed by a special form of an edge of type (2). P6 is not satisfied by the multivalued dependencies[4].

We can eliminate edges of types (3) and (4) if two kinds of vertices 0 and I are introduced, where 0 denotes an event which never occurs and I denotes an event which always occurs. An edge X_i——X_j of type (3) is represented by $I \longrightarrow X_i | X_j$. An edge X_i---X_j of type (4) can be replaced by $X_i \cup X_j \longrightarrow 0$ because of the above P9.

By these correspondences, P7 and P8 become special cases of P6 and P4, respectively.

A special case of P2 is expressed as follows.

P2': If $X_i \longrightarrow 0$ and $X_i \subseteq X_j$, then $X_j \longrightarrow 0$.

It is also a special case of P4.

3. Computation Methods

There are the following possible problems for subset graphs.

(a) The graph is consistent or not.

(b) Reduction of the number of edges, or minimization of the constraints.

(c) Determine whether a given restriction can be implied by a graph or not.

If a subset graph does not contain edges of type (3), then the graph is always consistent. The existence of edges of type (3) together with other types of edges causes inconsistency, i.e., there exists some A_i which is required to occur by one part of the graph while A_i is required not to occur by another part of the graph.

The problems above can be formulated by a Boolean equation shown as follows.

Let a_i be a Boolean variable corresponding to A_i. a_i is 1 if and only if A_i occurs. The Boolean variables for 0 and I are constant 0 and constant 1, respectively. Each $X_i \rightarrow X_{j1}|X_{j2}|\cdots|X_{jp}$ is expressed by $f(X_i)\bar{f}(X_{j1})\bar{f}(X_{j2})\cdots\bar{f}(X_{jp})$, where $f(X_j)$ is a logical product of all a_i's such that $a_i \in X_i$, and $\bar{f}$ is a logical negation of f.

For example, if $X_1=\{A_1,A_2\}$, $X_2=\{A_3,A_4\}$, $X_3=\{A_5,A_6\}$, the following correspondences are obtained

$X_1 \longrightarrow X_2$: $a_1a_2\overline{a_3a_4}$

$X_1 \longrightarrow X_2|X_3$: $a_1a_2\overline{a_3a_4}\,\overline{a_5a_6}$.

$X_1 \text{---} X_2$: $\overline{a_1a_2}\,\overline{a_3a_4}$ (because $X_1 \text{---} X_2$ is equivalent to $I \longrightarrow X_1|X_2$).

X_1---X_2 : $a_1a_2a_3a_4$ (because X_1---X_2 is equivalent to $X_1 \cup X_2 \longrightarrow 0$).

Let F be the logical sum of all the logic expressions corresponding to all the edges in the graph. Using F, the above problems can be solved as follows.

(a) The graph is consistent if and only if F=0 has a solution.

(b) The reduction problem corresponds to the reduction of the Boolean expression F.

(c) If we want to check whether $X_i \longrightarrow X_{j1} | \cdots | X_{jp}$, check whether $F=F+f(X_i)\bar{f}(X_{j1})\cdots\bar{f}(X_{jp})$.

By the results of the logic circuit theory and the relational database theory, we have efficient algorithms for the following cases.

(i) Detection of inconsistency when all edges are of types (1), (3) and (4) and every vertex corresponds to an event set consistent of one event (quadratic Boolean equations).

(ii) Minimization problem for graphs consisting of edges of type (4) only or edges of type (3) only (minimization of positive functions or negative functions).

(iii) Some minimization problems for graphs consisting of edges of type (1) only (minimum cover for functional dependencies [8])

(iv) The problem (c) when the graph contains only edges of type (1) (functional dependency problem[3])

(v) The problem (c) when the graph contains only edges of type (1) and edges of restricted version of type (2) (functional and multivalued dependency problem[2]).

When only edges of types (1),(3),(4) are used, the following properties hold.

P10: If $X_i \longrightarrow X_j$, $X_i \longrightarrow X_k$ and $X_j \longrightarrow X_k$
then X_k occurs (Fig.1).

P11: If $X_i \longrightarrow X_j$, $X_i \longrightarrow X_k$ and X_j---X_k
then X_i does not occur (Fig.2).

P12: If $X_i \longrightarrow X_j$ and $X_i \longrightarrow X_j$
then X_j occurs (Fig.3).

P13: If $X_i \longrightarrow X_j$ and X_i---X_j
then X_j does not occur (Fig.4).

These properties are easily proved, since any subgraph consisting of edges of type (1) satisfies that the label sequence of any directed path is $00\cdots011\cdots1$, where 0 and 1 on vertex X_i show that X_i does not occur and X_i occurs, respectively (Fig.5).

Using the above properties, reduction of a given subset graph is possible.

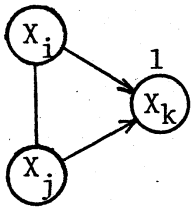

Fig.1 – P10

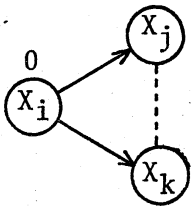

Fig.2 – P11

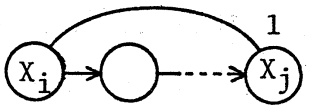

Fig.3 – P12

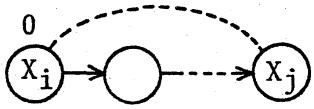

Fig.4 – P13

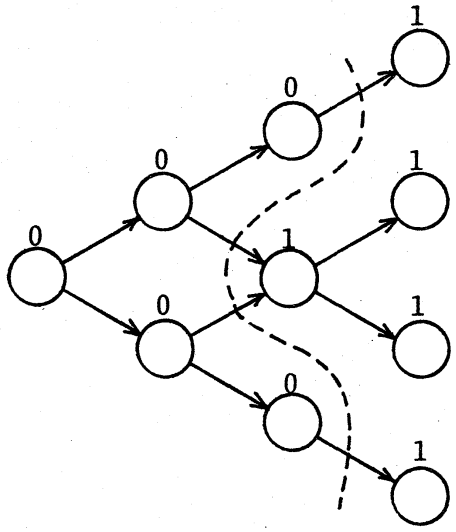

Fig.5 — A property of edges of type (1).

4. Labeled Subset Graphs and Their Application to an Authorization Mechanism

A subset graph introduced in this paper can be regarded as a kind of a dircted hypergraph. Like a labeled directed graph, we can define a labeled subset graph, where each edge is labeled by an integer. A label will mean a cost for the transition, the time of its creation, etc. Formally a labeled subset graph is defined as G(S,V,E,W), where W is a function from E to a set of integers and other symbols are the same as subset graphs.

As a practical example of a labeled subset graph, a new generalized authorization mechanism is discussed in this section.

In System R an advanced dynamic authorization mechanism is used, which was proposed by Griffiths and Wade[7] and modified by Fagin[6]. Key ideas of this GWF mechanism are as follows.

(1) Any user may be authorized to create a new file.

(2) If the user wishes to share his file with other users,

he may use the GRANT command to give various privileges on that file to various users.

(3) The user may grant a set of privileges with the grant option, which permits the grantee to further grant his acquired rights to other users.

(4) Any user who has granted a privilege may subsequently withdraw it by issuing the REVOKE command.

One important problem is to calculate all users who will lose the privilege by a given revoke command, since if user i loses the privilege, all the users who have gotten the privilege from i will also lose the privilege unless they got the same privilege from other users. This problem is not simple because of its recursive nature.

The mechanism can be represented by a labeled directed graph, where each user corresponds to a vertex and a privilege transfer from i to j at time t is shown by an edge $e_{ij}(t)$ from i to j, labeled by t.

A problem of the original model[7] was pointed out by Fagin [6]. It can be solved if all the grants which are issued from the same grantor to the same recipient at different time are recorded. This method, however, causes another problem since the number of edges is unlimited even for a finite number of users.

We will permit a new kinds of grant transfers to solve this problem. Grants which are issued from the same grantor to the same recipient at different time are usually required when the grantor wants to make sure that the recipient has the privilege

whenever the grantor has it. In order to represent it, we will introduce a new label * for edges. If $e_{ij}(t)$ is labeled by *, it is regarded as $e_{ij}(t')$ for any $t' \geq t$. This method will be used in the simple threshold authorization mechanism discussed below.

In computer systems, usually several security levels are required, such as 'top secret', 'secret', 'confidential', and 'public'. According to the security levels of files, we should have different levels of authorization mechanisms. The GWF mechanism cannot handle such a security level. In order to handle this problem, we can use labeled subset graphs in stead of labeled directed graphs in the above discussion. Subset graphs correspond to the new mechanism use only edges of type (1) and thus we have an efficient procedure to calculate the effect of a revoke command.

A simple threshold authorization mechanism is defined as follows.

$$A_{ST}(F,P) = (U, G, U_o, Q)$$

It is defined for each combination of F (a set of files with which the mechanism is concerned) and P (a set of privileges). $U = \{u_1, u_2, \ldots, u_n\}$ is a set of users. $G = \{g_1, g_2, \ldots, g_m\}$ is a set of granting operations, where $g_i: U_i \rightarrow u[t,w]$ and U_i is a set of users ($U_i \subseteq U$), $u \in U$, t is a time stamp for this granting operation, and $w \in \{r, g, r^*, g^*\}$ shows the type. r and g are granting privileges and granting privileges with grant option, respectively. * shows the grant transfer such that the grantor wants to make sure that the recipient has the privilege

whenever the grantor has it. U_o is the set of users called the co-owners of F. $Q = \{q_r, q_g\}$ is a set of threshold values. q_r and q_g correspond to these of the granting privileges and the granting privileges with grant option, respectively.

For example, g_1: $\{u_1, u_2\} \rightarrow u_3[t, r(\text{or } r^*)]$ ($u_3[t, g(\text{or } g^*)]$) shows that users u_1 and u_2 together grant privileges P (with grant options, respectively) to u_3 at time t. The number of users in the left side of $\rightarrow$ is determined by $\min(q_r, |U_g|)$ ($\min(q_g, |U_g|)$, respectively). Here, U_p, U_g and U_n are subsets of U, each of which corresponds to a set of users with the privileges, a set of users with the privileges and grant transfer rights or a set of users without any rights at all, respectively. Since users in U_g have the privileges, $U_g \subseteq U_p$ and $U = U_p \cup U_n$.

For example, let $U = \{u_1, u_2, u_3, u_4, u_5\}$, $G = \{g_1, g_2, g_3\}$ and $U_o = \{u_1, u_2\}$ and $Q = \{2, 2\}$. Granting operations are assumed to be as follows.

g_1: $u_1u_2 \rightarrow u_3[10, g]$

g_2: $u_1u_2 \rightarrow u_4[10, r]$

g_3: $u_2u_3 \rightarrow u_4[20, g]$

The sets U_p, U_g and U_n are as follows.

	U_p	U_g	U_n
At time 5	u_1u_2	u_1u_2	$u_3u_4u_5$
At time 10	$u_1u_2u_3u_4$	$u_1u_2u_3$	u_5
At time 20	$u_1u_2u_3u_4$	$u_1u_2u_3u_4$	u_5

At least one user in the left set of g_i can revoke the granting operation g_i ($i = 1, 2, 3$). If u_2 revokes g_3 at time 30, u_3 will

lose the grant transfer right. If, however, u_2 revokes g_1 at time 30, u_3 will lose the privilege as well as the grant transfer right. Because of this, g_3 becomes ineffective and u_4 will lose the grant transfer right.

Because of the generalization, the calculation to compute an effect of one revoke command becomes complicated. We have the following efficient algorithm for the calculation.

[An algorithm to compute the effect of one revoke command]

(1) Let t_{ri} and t_{gi} be the time when u_i first obtained the privileges and the privileges with grant option, respectively. If not defined, a sufficiently large number is assigned.

(2) Assume that user u_i revokes g_j: $u_j \rightarrow u_{gj}[t_j,w]$ satisfying $u_i \in U_j$, renove g_j from the set G.

(3) Let U_{go} be the set of users whose t_{gi} is less than t_j. Users in U_{go} have the privilege as well as the grant transfer right and are not influenced by the revocation of g_j.

(4) If there exists g_k: $U_k \rightarrow u_{gk}[t_k,g]$(or $[t_k,g^*]$) such that $U_k \subseteq U_{go}$ and for any $u_{ki} \in U_k$, $t_{gk} < t_k$ (the condition is not required for g*), then u_{gk} is added to U_{go} and $t_{g_{gk}}$ is defined to be t_k(for g*, max(t_k, $t_{g_{ki}}$ for $u_{ki} \in U_k$)).

(5) Repeat step (4) until no new user is added to U_{go}.

(6) After calculating U_{go}, calculate U_{ro} as follows. U_{ro} is initially U_{go}. If there exists g_h: $U_h \rightarrow u_{gh}[t_h, r]$ (or $[t_h, r^*]$) such that $U_h \subseteq U_{go}$ and for any $u_{hi} \in U_h$, $t_{g_{hi}} < t_h$(the condition is not required for r*), then u_{gh} is added to U_{ro} and $t_{r_{gh}}$ is defined to be t_h(for r*,

$\max(t_h, t_{g_{hi}}$ for u_{hi} $U_k))$. Repeat this process until no increase of U_{ro} occurs.

(7) If there exists g's with w = r or g which are not used in steps (4) and (6), remove them from G. Granting operations with w = r* or g* which are not used in these steps, need not be modified.

5. Other Database Applications

There seems to be many areas where the concept of the subset graphs can be utilized. Possible applications to databases are listed as follows.

(1) Trigger system: Trigger is used in a database system to keep the integrity of the system automatically. For example, if a salary value is modified, we need to change values like average salary, company's profit, etc. The meaning of an edge $X_i \rightarrow X_{j1}|\ldots|X_{jp}$ is that if all events in X_i occurs, then for at least one X_{jk} all events in X_{jk} must be processed. Edges of types (3) and (4) give some constraints on the trigger computation. By a subset graph model we can detect inconsistency of a trigger definition and find all the processes to be executed.

(2) View computation: Relations stored in a database system are called base relations. Each user can generate his own relations (called views) by combinations of the base relations. There are also views generated by other views. One view can be generated by several different ways. If we assume

that the meaning of an edge $X_i \rightarrow A_j$ labeled by w is that the view A_j is formed by relations in X_i using cost w, the minimum cost view generation problem can be solved by the minimum path problem of a subset graph. A_i---A_j means two views A_i and A_j are not permitted at the same time.

The same model can be used to find the minimum computation solution to calculate some constraint. The cost above can be computation time in the case of centralized database systems and data transmission time in the case of distributed database systems.

Because of the similarity between the edges of type (2) and the multivalued dependencies, there may be efficient procedures for some subset graph problems. The author believes that there are many application areas of subset graphs besides databases.

Acknowledgement

Thanks are due to Professor S. Yajima for his discussions and Mr. M. Yoshikawa for his help in preparing the final manuscript.

References

[1] Armstrong, W.W. "Dependency structures of data base relationships." Information Processing 74. North Holland, Amsterdam, pp.580-583, 1974.

[2] Beeri, C. "On the membership problem for functional and

multivalued dependencies in relational databases." ACM Trans. on Database Syst. 5, 3, pp.241-259, Sept. 1980.

[3] Beeri, C., and Bernstein, P.A. "Computational problems related to the design of normal form relational schemas." ACM Trans. on Database Syst. 4, 1, pp.30-59, March 1979.

[4] Beeri, C., Fagin, R., and Howard, J.H. "A complete axiomatization for functional and multivalued dependencies in database relations." Proc. 3rd ACM SIGMOD Int. Conf. Management of Data, Toronto, pp.47-61, 1977.

[5] Fagin, R. "Functional dependencies in a relational database and propositional logic." IBM J. Res. Dev. 21, 6, pp.534-544, Nov. 1977.

[6] Fagin, R. "On an authorization mechanism." ACM Trans. on Database Syst. 3, 3, pp.310-319, Sept. 1978.

[7] Griffiths, P.P., and Wade, B.W. "An authorization mechanism for a relational database system." ACM Trans. on Database Syst. 1, 3, pp.242-255, Sept. 1976.

[8] Maier, D. "Minimum covers in the relational database model." Journal of the ACM, 27, 4, pp.664-674, Oct. 1980.

[9] Sagiv, Y. "An algorithm for inferring multivalued dependencies that works also for a subset of propositional logic." Tech. Rep., Dep. Computer Science, Univ. Illinois at Urbana-Champaign, 1979.